Towards Adaptive Covert Communication System

Fedor V. Yarochkin, Shih-Yao Dai, Chih-Hung Lin *, Yennun Huang*, Sy-Yen Kuo
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
{sykuo@cc.ee.ntu.edu.tw}

Abstract

Covert channels are secret communication paths, which existance is not expected in the original system design. Covert channels can be used as legimate tools *of censorship resistance, anonimity and privacy preservation to address issues with "national" firewalls, citizen profiling and other "unethical" uses of Information Technology. Current steganographic methods that implement covert channels within network traffic, are highly dependent on particular media data or network protocol to hide data. In this paper we investigate the methods and an algorithm for implementing adaptive covert communication system that works on real-world Internet, capable of using multiple application-level protocols as its communication media and can be implemented as network application, therefore requires no system modifications of communicating nodes. The key difference from previous solutions is the use of adaptive redundant mechanism, which allows real-time underlying protocol switching and adaptation to the dynamic network configuration changes.*

Keywords: self-adaptation and self-organization, covert-channels, steganography

1. Introduction

Censorship and information access control is one of common human right violations, comitted by oppressive political regimes or abusive corporations. Therefore, the clandestine communication methods and communication methods that allow communication parties to preserve secrecy and anonymity has been of a certain interest recently in academic community[18],[2],[8],[6], [13].

However these implementations relay on a *single* network or application protocol, such as HTTP/HTTPS[8],ICMP[14], TCP/IP[11], DNS[3] or SMTP[5], to build communication links. Therefore in censored network it would be enough to restrict or block given protocol to effectively defeat the covert channel communication ability. Also, such implementations as [6] and [13] sacrifice protocol performance and robustness in favour of preserving secrecy. We think that in real-world Internet such degree of secrecy is unnecessary, as the majority of Internet cencorship implementations are implemented using automated tools.[16][10]

Therefore there should be an option for the covert channel user to be able to choose the degree of secrecy vs. performance, that she may desire.

In particular, in the past few years there was obvious interest in building robust and high-rate covert channels using existing network infrastructure[17]. However these designs still suffer from the same problem of relaying on single network protocol to communicate. Therefore we see the flexibility and adaptability to the dynamic network changes as one of the key requirements to the robust, high-performance covert communication channel systems.

Additionally it is extremely important to be able to detect and recover possible communication errors, which may occur due to intermediate nodes malfunctioning or malicious activity. Therefore we propose a network communication framework that implements network discovery function, adaptive mechanisms with pluggable network protocols architecture and optional voting mechanism for error detection in order to cope with dynamic network configuration changes while maintaining persistency of covert communication channels.

These channels,when implemented with error detection mechanism, include redundancy options, so the transmission of equivalent data is performed over a number of underlying channels (variety of protocols and protocol configurations). And the voting mechanism used to identify possible data alteration or communication error.

The adaptive mechanisms are also designed to detect possible interruptions in the communication channel, which

*Chih-Hung Lin and Yennun Huang are with Institute for Information Industry, Taipei, Taiwan

may be caused by changes within hostile network environment (such as firewall or network configuration changes), and are capable of real-time adapting the network communication media to re-establish communication link. An architecture adopting diverse redundant network protocol stacks is proposed for that purpose.

The rest of the paper is organized as follows: Section 2 presents background information and introduces proposed communication protcol, Section 3 discusses detectability of proposed communication protocol. Section 4 discusses advantages and disadvantages of proposed model. Section 5 discusses implementation issues and performance evaluation, Section 6 makes conclusive remarks, Section 7 proposes several ideas for future work and Section 8 includes references to web resources, where discussed implementation code shall be available.

2. Communication Protocol

2.1. Threats and Model

Any network is a dynamic environment, which does not guarantee persistancy of its nodes. One of the key issues, which a covert communication protocol would have to deal with is dynamic network configuration changes, such as network routing changes, firewall configuration changes and so on. Therefore a reliable covert communication protocol must be able to detect such configuration changes and be capable of adapting to it in order to preserve communication link.

On the other hand, the framework must avoid using network protocols, which may appear to be unusual within operating network environment, as such protocols are most likely to be blocked on firewalls, and attempts to communicate using these protocols might be detected by Intrusion Detection Systems (IDS), which use anomaly detection mechanisms to detect suspicious activities.

Our model would look as follows: we have two communicating agents A_1 and A_2 with subset of communicable protocols P_1 and P_2 respectevely. Agent A_1, in order to communicate with Agent A_2, must detect a subset of communicable protocols P_1 so that $P_1 \cap P_2$ (protocols, which can be received by Agent 2) and this subset of $P_1 \cap P_2 \notin P_b$, where P_b is set of protocols, which are not routed, or blocked by the network environment.

In proposed framework, the "network environment learning" phase is focused on learning the subsets of P_i for each communicating agent, while The protocol handshake is the process of finding such intersection $P_1 \cap P_2 \Rightarrow P_1 2$ such that $P_1 2 \notin P_b$.

While designing this framework we focus on reliability of the communication framework that uses covert channels, embedded into different protocols to communicate. We do not focus on detectability of each protocol covert channel implementation on the moment [1]

2.2. Protocol Design

A protocol is a distributed algorithm that solves a problem in a distributed system[7]. In our context we propose adaptive covert channel communication mechanism using a set of protocol modules which are distributed across the machines. The set of protocol modules are organized in protocol stack. Each of the protocol modules is designed to embed communication data within one of the existing network protocols (plain UDP, plain TCP,DNS, SMTP and so on) in a manner, so that communication channel would not be detected using anomaly detection techniques.

The decision was taken to use application-level protocols in order to be able to tunnel traffic through application-level gateways (such as XMPP servers,SMTP hosts, DNS query forwarding services, HTTP proxies and so on). Methods of forwarding traffic is not within the scope of this paper. These methods are usually specific to each protocol and shall be only briefly discussed with references for further information.

The organization of protocol modules in stack is not strictly layered, instead, the protocol modules can cooperate with other protocol modules in the same stack. The execution of the protocol modules is driven by events.

Each of the protocol modules has one or more handlers, where each of the handlers is designed to handle a particular event. When event occurs, all handlers that are bound to this event are executed. Priority is used to control the execution process.

The protocol execution is divided into 2 basic phases:

- Network Environment Learning Phase
- Communication Phase

During the Network Environment Learning Phase (NEL) each of the agents A_i is learning about used protocols by passively monitoring network traffic on one or more network interfaces. Identified protocols P_i are matched to the subset of protocols p_i, which each of the agents is capable of communicating with (has modules, which support these protocols), and only a subset of $P_i \cap p_i$ will be used in communication.

Once the initial information gathering phase (NEL) is completed, agents are ready communicate. NEL process does not stop at this point and continues to monitor network environment in order to identify possible changes in network protocol usage patterns.

[1] Covert channels within the Internet standard protcols have been analyzed in [1], [12], [15], in this paper we used simplified implementations for XMPP, TCP on arbitrary port, and UDP on arbitrary port protocols.

Suppose we want agent A_1 to communicate with agent A_2. The *handshake process* takes place in order to establish communication link between A_1 and A_2 before A_1 and A_2 can exchange the data. During this process the agent A_1 uses one of the protocols from subset $P_i \cap p_i$ to attempt to establish the communication.

The loaded protocol module sends requests, a sequence of packets which identify communication attempt using specific protocol modules.

One or more protocols from the subset $P_i \cap p_i$ can be activated during initial handshake. In this case a *burst* method may be used to collect and transmit these packets simultaneously.

Some of the requests, which belong to set of "blocked" protocols P_b will be discarded by the network environment and only $P_1 \notin P_b$ requests will reach the agent A_2.

One of the goals of NEL phase is to minimize number of packets blocked during the communication attempts, as this may unveil the secret communication attempt.

If two or more packets reach the agent A_2, the agent loads the protocol modules, which it has available for communication $P_2 \cap p_2 \notin P_b$ and issues the responses. Confirmation from A_1 on used protocol module is expected to select one of the protocol modules for communication and start actual data exchange process.

Round-Trip Time (further RTT) is calculated when response is received. This RTT value is noted for each of the protocols p_i and later used to detect connection drops, packet drops and network configuration changes.

Data transmission process - during this process the actual data transmission takes place. The data is transferred using the selected data transmission module. The keep-alive queries are sent on fixed interval to ensure that communication channel over selected protocol is still available. In case if keep-alive response time exceeds $RTT * 2$, the rehandshake phase is initiated. During this phase the communication sides are to negotiate new protocol module, which should be different from the modules, used at the previous step.

A success score is maintained for each of the protocol modules p_i and this score is used in empirical formula, which selects one of possible protocols for communication during the handshake process.

2.3. Covert Operation

The proposed protocol allows two nodes to communicate by embedding communication data into the application-level protocols, which are observed to be used within existing network environment. This method is much more efficient than simple tunneling of the communication data within network layer protocol, as this would allow parties to communicate even if no direct network connection is possible between two communicating agents.

2.4. Protocol modules grouping

Data transmission modules maybe grouped in particular groups so only one member of a group can be selected for communication process at any time. This is done to avoid having data transmission modules that use same or similar underlying protocol for data transfer, as this may affect efficiency of the voting mechanism check.

2.5. Design of Protocol Modules

Methods, using which the actual data transmission modules are implemented, are not restricted by any of existing protocol specification. For example DNS packet payload may be used as data transmission medium, padding in Ethernet frames may also be utilized, when communicated party is connected directly using Ethernet cable.

More details on embedding covert communication data into Internet protocols could be found in [1], [12], [15][9].

2.6. Design of Adaptive Algorithm

A very simple, score-based adaptive algorithm is proposed for the current implementation, which is shown on Figure 1. The algorithm uses the protocol module "success score" to select next communication module from the list of available protocol modules. The "success score" of each protocol module represents estimated probability of how successeful each protocol module would be to establish connection. The "success" rate is calculated using empirical formula that uses protocol statistic data from network monitoring component (to obtain list of currently active network protocols) and historical record of modules, which were, or were not successefully used in previous connections.

Random factor is introduced to provide algorithm with the capability of identifying network changes, which could not be observed from the network traffic monitoring data and historical listing of previously successeful modules.

Obviously, it should be possible to use other, more complicated adaptive methods (for example those, which would utilize Neural Networks) in place of currently suggested algorithm. These algorithms could be the subject of our further research.

2.7. Error Detection mechanisms

It might be possible to implement communication quality detection mechanisms, which could detect a subclass of communication errors or attacks against communication channel

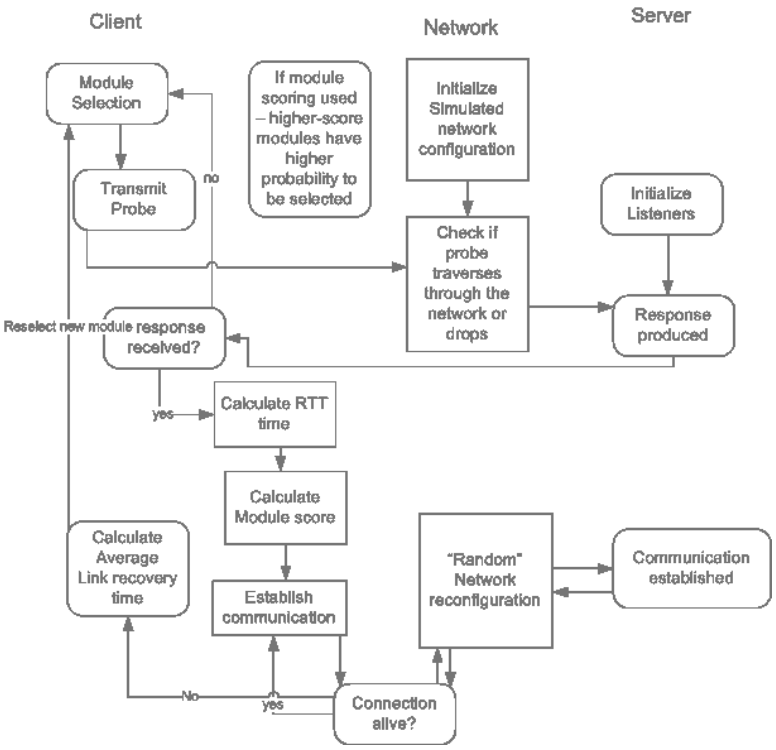

Figure 1. Implementation Diagram

For example a voting mechanism and simultaneous multi-protocol communication can be used to detect possible data alteration due to communication errors or man-in-the-middle attacks. This mechanism would use 3 protocols to transmit data simultaneously and the voting mechanism would be used to identify whether possible data alteration took place.

Communication link quality detection mechanism could be implemented to detect various low-level communication errors, such as connection timeout errors, routing errors and so on. Once communication error is detected, the system may be required to re-negotiate communication protocol.

3. Detectability and Robustness of the Covert Operation

The proposed approach makes it harder to detect covert communication channels using standard signature based intrusion detection systems, which target at picking up communications at certain ports or protocols with certain payloads. The volatility of the protocol modules makes it possible to dinamically update sets of communicable protocols, if communication over certain protocols was detected and blocked.

The protocol normalization based detection techniques, such as mentioned in [4] may detect particular covert channel communication that uses some particular protocol. However even if the detected protocol is identified and blocked by ”the authority”, this would not have critical impact on the communicability of the framework as long, as other communication protocols are available.

4. Advantages and Disadvantages of Adaptive Protocol

Reliability of the data transmission process is the primary advantage of using such protocol. When the framework is in use, communication between two given parties does not depend on actual communication medium implementation. Such mediums can be loaded and unloaded dynamically and adapt to changing environment.

The complexity and data overhead within the underlying application-level protocols are obvious disadvantages of applying such framework, but it is known fact that there s no win-win solution in security, and data overhead is being traded for additional reliability features of the communication link.

Additionally, intelligent (instead of random) selection of protocol modules (based on success score) may significantly minimize data overhead in the communication pro-

cess and ”noise” on the network layer.

5. Performance evaluation

To evaluate the performance of the proposed covert channel communication system, the proposed algorithm was implemented in C code with loadable modules for each of the network protocols.

Three protocols were selected for performance evaluation testing: XMPP protocol, TCP protocol (arbitrary port) and UDP protocol (arbitrary port). The statistical data below was collected using implementation of TCP connection forwarding on a number of different ports, which represents different protocol modules in proposed algorithm.

The basic outline of implemented software component was demonstrated on the diagram (Figure 1). The tested network environment was implemented within two physical Ethernet segments segregated by a router in the middle. The testing process starts with initializing the network environment, which would block a set of TCP and UDP protocols on the router. A script was used so simulate legimate communication between two nodes on arbitrary port. The purpose of evaluation was to verify that the system is capable of identifying usable protocols within the network environment and establish the communication link between two nodes.

”Random” configuration changes on the router were introduced to change network environment settings every N number of seconds.

5.1. Results

During the evaluation process, the network configuration changes occurred 200 times forcing the covert channel to reestablish active connection. The connection recovery time, number of failed attempts per successeful connection and data overhead for different network protocol modules were calculated and are presented on the following diagrams.

During the testing process, the client side algorithm was executed in two modes:

- With Adaptive Algorithm. The implemented Adaptive Algorithm used simple protocol module success score, which signifies how ”successeful” connection with this protocol module could be. This ”success score” is calculated using current network traffic monitoring snapshot and statistics of previous successeful connections. The result is presented on Figure 2 and 3. The algorithm keeps track of successfully used modules by use of ”success” score, and the score affects probability of a module being selected during next step.
- The results of algorithm execution when protocol modules were selected randomly, is presented on Figure 4

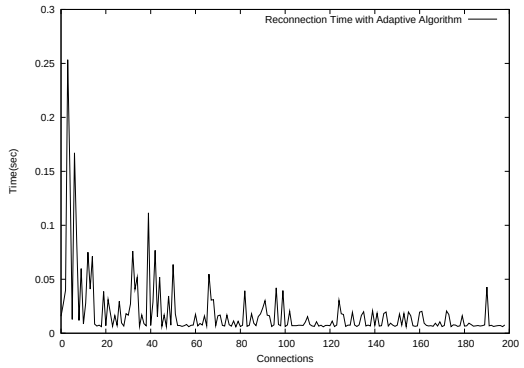

Figure 2. Reconnection Time for test phase with adaptive algorithm

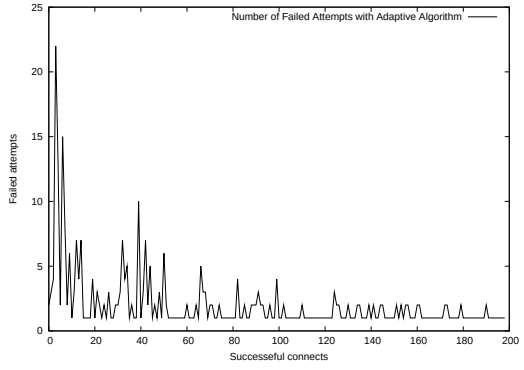

Figure 3. Number of failed connect attempts per successeful connect for test phase with adaptive algorithm

and 5. In this case algorithm selects modules on random basis. It should be noted that handshake process was observed to be much slower when Application-level protocol (XMPP) was used.

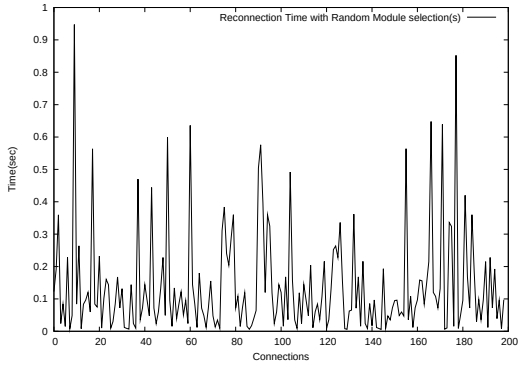

Figure 4. Reconnection Time for algorithm testing with random protocol module selection

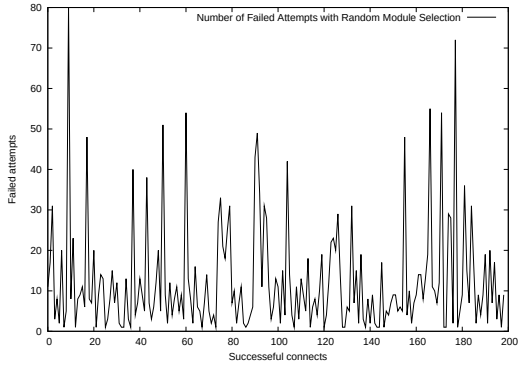

Figure 5. Number of failed attempts per successful connect for algorithm testing with random protocol module selection

As it could be observed from these diagrams, the application of adaptive protocol sufficiently reduces reconnection time and minimizes the number of failed attempts per successful connection, which also should overall decrease detectability of covert communication system.

The estimated connection overhead values were calculated by taking average amount of transmitted "protocol" data that has to be transmitted along with original data frame to ensure the frame delivery to the target.

Estimated Connection Overhead	
Protocol	Average Overhead (bytes/frame)
XMPP	over 100 bytes (XML)
Plain UDP	28-30 bytes
Plain TCP	40 bytes
ICMP echo	28-30 bytes

6. Conclusion

In this paper we proposed a different approach to implement adaptive covert communication channel that is abstracted from physical network protocols, which carry actual data payload. Use of frameworks like this may allow creative use of existing network protocols to perform data transmission as such data may be placed in unused, or reserved fields. If data corruption occurs over particular links (i.e. links that may actually utilize selected fields), such protocol modules may be simply dropped without fear of losing communication. Use of such communication frameworks would also allow to create network stacks, that dynamically adapt to changing network environment.

7. Further work

Point to Point (P2P) communications have been an emerging trend in recent few years. Suggested covert communication channel algorithm may also be used in context of point to point communication. Interesting research direction would be to design message routing and message forwarding algorithms for nodes to communicate reliably in P2P fashion.

Adaptive algorithms for communication protocol module selection could be an interesting theme for additional research. Apparently different adaptive algorithms may perform differently within different network environments.

8. Availability

Developed application is free software, released under GNU General Public License and will be shortly available via HTTP from

`http://o0o.nu/rebounce`

Acknowledgement

This study is conducted under the "III Innovative and Prospective Technologies Project" of the Institute for Information Industry which is subsidized by the Ministry of Economy Affairs of the Republic of China.

References

[1] M. Bauer. New covert channels in HTTP: adding unwitting Web browsers to anonymity sets. In *Proceedings of the 2003 ACM workshop on Privacy in electronic society*, pages 72–78, 2003.

[2] S. Gianvecchio. Model-Based Covert Timing Channels: Automated Modeling and Evasion. In *Proceedings of Recent Advances in Intrusion Detection (RAID)*, 2008.

[3] T. Gil. IP-over-DNS using NSTX. http://thomer.com/howtos/nstx, 2005.

[4] Z. Kwecka. Application Layer Covert Channel Analysis and Detection. In *Technical Report, Napier University Edinburgh.*, 2006.

[5] M. Lundstrom. MailTunnel - IP over SMTP tunnel. http://gray-world.net/tools/mailtunnel-0.2.tar.gz, 2000.

[6] M. E. E.-Z. Magdy Saeb, Eman El-Abd. On covert data communication channels employing DNA recombinant and mutagenesis based steganographic techniques. In *Proceedings of the 2007 annual International Conference on Computer Engineering and Applications*, Jan. 2007.

[7] A. S. Oliver Rutti, Pawel T. Wojciechowski. Service interface: a new abstaction for implementing and composing protocols. In *Symposium on Applied Computing, Proceedings of the 2006 ACM symposium and Applied Computing*, 2006.

[8] P. S. Roger Dingledine, Nick Mathewson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th Usenix Security Symposium*. Usenix Association, 2004.

[9] P. B. S. Zander, G. Armitage. Covert Channels in the IP Time To Live Field. In *Proceedings of Australian Telecommunication Networks and Applications Conference (ATNAC)*, 2006.

[10] P. B. S. Zander, G. Armitage. A Survey of Covert Channels and Countermeasures in Computer Networks Protocols. In *IEEE Communications Surveys and Tutorials*, pages 9(3):44–57, 2007.

[11] P. B. S. Zander, G. Armitage. Covert Channels and Countermeasures in Computer Networks Protocols. In *IEEE Communications Magazine*, pages 45(12):136–142, 2007.

[12] C. S. Serdar Sabuk, Carla E. Brodley. IP covert timing channels: design and detection. In *Proceedings of the 11th ACM conference on Computer and Communication Security*, pages 178–187, 2004.

[13] A. E. Song Li. A Network Layer Covert Channel in Ad-hoc Wireless Networks. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, Oct. 2004.

[14] D. Stodle. Ptunnel - Ping Tunnel. http://www.cs.uit.no/daniels/PingTunnel, 2005.

[15] Z. Trabelsi. A novel covert channel based on the IP header record route option. In *International Journal of Advanced Media and Communication*, pages 328–350, 2007.

[16] S. G. H. Wang. Detecting Covert Timing Channels: An Entropy-Based Approach. . In *Proceedings of 14th ACM Conference on Computer and Communication Security(CCS)*, pages 9(3):44–57, 2007.

[17] J. C. M. William K. Geissler. Exploiting error control in network traffic for robust, high rate covert channels. *International Journal of Electronic Security and Digital Forensics*, 1(2):180–193, Jan. 2007.

[18] R. K. C. C. X. Luo, E. W. W. Chan. TCP Covert Timing Channels: Design and Detection. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2008.